AMENDMENTS TO THE SPECIFICATION:

Please replace paragraph 0019 of the published application with the following amended paragraph:

USP 5,732,267 (Smith) provides for "Caching/Prewarming Data Loaded From CD-ROM." Data defining pages and objects of a multimedia work are transferred in the background to minimize delays that would otherwise be incurred. In playing a multimedia work that is recorded on a CD-ROM, a personal computer (10) that includes a central processing ~~using~~ unit (CPU) (23) transfers data for selected pages and for objects on a page of the multimedia work into a cache, using free CPU cycles, so that the data are available when needed. This technique is particularly useful in transferring data for animation objects of a multimedia work, since it enables two animations to play concurrently without incurring a delay to load the data for the second animation when the page is loaded and avoids interrupting the execution of the first animation at the time that the second animation must start executing.

Please replace paragraph 0026 of the published application with the following amended paragraph:

Figure 1 illustrates the Push-Pull Gateway (hereafter iPPG ~~or iPPG~~) End-to-End (E2E) system used to implement the present invention.

Please replace paragraph 0034 of the published application with the following amended paragraph:

Figure 1 illustrates a Push-Pull Gateway (hereafter iPPG ~~or iPPG~~) End-to-End (E2E) system **100** used to implement the present invention. This Push-Pull Gateway system is described in greater detail in co-pending application entitled "System and Method Providing a Push Gateway between Consumer devices and Remote Content Provider Centers". The system components (to be described below) of the iPPG collectively achieve the Push, Pull, and send features of the gateway (iPPG). In Figure 1, the remote **102** or local **103** application service providers (ASPs) submit (or Push) contents, over a network N (e.g., the Internet) via a protocol such as HTTP, to the iPPG **104**. The iPPG **104** is able to either accept or reject such requests by ASPs **102** and **103**. The iPPG is also able to retrieve (or Pull) contents from data server **105** as selected by a station operator. The iPPG of the present invention, with the help of an operation administration module (OAM) **110**, prioritizes, schedules, and sends

datagrams to the radio transmitter station or iExciter (exciter **106**) over interface **E**. Receiver **108** (client) acquires the data and using turbo broadcast layer **113** de-encapsulates the data. The data is then displayed on terminal **114**. Furthermore, a billing procedure keeps track of all data pushes (via pre-defined logistics **112**) from various ASPs for billing purposes. As will be detailed later, when in listen mode, the data receiver **108** displays the received data continuously, or, upon demand, as per filtration activated by subscriber.

Please replace paragraph 0039 of the published application with the following amended paragraph:

A scheduler **324**, then parses control entity of the message and determines time/schedule for contained instructions and passes such information for storage on to push recorder **322**. If the instruction extracted by the scheduler **324** includes retrieving data, the content fetcher ~~**326**~~ **323**, in conjunction with the scheduler **324** and a network database **328**, pulls data from content providers **330** via a network **332**, such as the Internet. The pulled data is then transformed and encoded (via data transformer **334** and encoder ~~**336**~~ **333**, respectively) into a format requested by the client. Furthermore, data transformer **334** and encoder **333** split the data into octet data blocks, assign serial numbers to all packets, and pass them on to addressing module **342** and cache **338**. Lastly, the data from the addressing module is passed onto the IBOC outbound queue **344** to various end devices linked to a broadcast network **343**, such as an IBOC network.

Please replace paragraph 0046 of the published application with the following amended paragraph:

Thus, in summary, the iPPG ~~or iPPG~~ is able to push data from various content provider centers and is also able to pull data from remote content providers. The content provider centers and remote content providers are able to communicate with the iPPG via a network (LAN, WAN, Internet, etc.). Based upon the request from the content provider centers, the data is then pushed via a network such as an IBOC network onto various end devices (clients).

Please replace paragraph 0047 of the published application with the following amended paragraph:

It should be noted that although only one iPPG ~~(or iPPG)~~ is described, one skilled in the art of networked communication can envision using multiple iPPGs ~~(or iPPGs)~~, for

distributed processing, wherein such gateways are controlled by one or more centralized gateways. Thus, one skilled in the art can envision using various combinations including, but not limited to, one iPPG and many transmitters, a set of networked iPPGs, and a master iPPG and a scaled down iPPG. Furthermore, although the iPPG, remote content providers, and content provider center are shown to be separate entities communicating over various networks, one skilled in the art can envision them as being implemented locally in one single entity.

Please replace paragraph 0049 of the published application with the following amended paragraph (**this paragraph has previously been amended**):

Figure 4 illustrates how incoming data is handled at the client (receiver's end - an IBOC-enabled mobile device **400**). An antenna **401** located on the receiver first receives incoming data, and detection equipment **402** detects such data and optionally amplifies the signal. The received data is then deinterleaved via deinterleaver **405**, demodulated via demodulator **406**, decoded via a transport decoder **407** (such as a iDAB transport layer decoder), and further decoded via a data link layer decoder **408**. If data is audio, it is forwarded to PAC decoder **419**, and if it is meant for turbo broadcast layer, it is forwarded to **408**. Audio signals are converted into audible sounds and are forwarded to the speaker **403**. The detection equipment **402** uses a channel quality measurer **404** to calculate the quality associated with a tuned channel. It should be noted that the host processing unit **409** actively controls the above-described deinterleaver, demodulator, decoder, and turbo broadcast layer decoder. Lastly, the host processing unit 409 and associated memory ~~**410**~~ process the decoded data before being presented to the end user device, via a display device **412** (with OEM I/O input **411**).

Please replace paragraph 0063 of the published application with the following amended paragraph:

According to the determined time schedule, and in the case of a pull request submitted at step **502**, the content fetcher, at step **520**, establishes a session with appropriate server on remote network area and retrieves the requested data files. Content provider may submit contents to be pushed at step **503**. Step **506** is used for authentication and registration of the content provider. At step **508**, contents can be pushed real-time (if bandwidth is available) or it can be scheduled for a pre-download later. Even if bandwidth is available pre-download is recommended (may be at a lower cost). If no pull request has been submitted at step **502** or

after completion of step **520**, the pushed/pulled data is passed on to data transformer/encoder at step **522**. If the data submitted to data transformer/encoder needs to be transformed into a suitable mark-up language **524** for consumer device(s), the data transformer/encoder effects such data transformation by the use of translation software in step **526**. Then, at step **530**, TBL-SSAL splits the data into multiple octet data blocks, assigns identical serial numbers to all those packets, and passes them on to the cache and addressing module. In step **532**, the addressing module ~~the~~ parses the control entity of the push/pull message for addressing instructions.